ILLINOIS INSTITUTE OF TECHNOLOGY

HACKERS, WILD DUCKS, AND SKUNK WORKS:

TIMESHARING AND VIRTUALIZATION FROM 1959 TO 1968

ITM 554: OPERATING SYSTEM VIRTUALIZATION

JEREMY HAJEK

BY

SEAN P. MCBRIDE

CHICAGO, IL

30 MARCH 2013

AD MAIOREM DEI GLORIAM

In Memoriam Robert Jay Creasy (1939-2005)

Thanks for inspiring Wild Ducks everywhere!

Please consider this your Twenty-One Quack Salute.

Have you ever heard of ENIAC, SAGE, or System/360? If you are a budding technology historian or have ever visited the Computer History Museum, then perhaps you have. What about the Macintosh? Chances are that, even if you are not an Apple fanatic, you know more about the Macintosh than ENIAC, SAGE, or System/360, despite the fact that they are all of galactic importance in the history of computing. ENIAC was the first fully electronic computer, and it ran a one million punch card ballistics test for the hydrogen bomb. SAGE had the ability to shoot down Soviet nukes, used Light Pen technology that resembled something out of Star Wars: A New Hope, and cost the American taxpayers billions of dollars.[1] The System/360 neither launched nor intercepted nukes, but established the first compatible family of computers, laying the groundwork for the software industry. Don't feel bad if you didn't know this! The reason that you remember the Mac and not ENIAC, SAGE, or System/360 can be explained by human psychology. Because the Macintosh is a tool for a single end-user, its use likely activates the same sorts of neural pathways that fired when *homo erectus* crafted and wielded his first spear. Despite the fact that we form societies and build shared infrastructure, such as aqueducts, highways, or supercomputers, humans are preprogramed to prefer personal things. This is why Computer History Museum visitors powerwalk past Abacuses, Hollerith machines, ENIAC, SAGE, IBM mainframes, Cray supercomputers, and PDP minicomputers, yet linger by the 1969 Nieman-Marcus Kitchen Computer (Computer History Museum n.d.) and the section with PCs and Atari video games. To put it bluntly, nearly all laymen and a large share of technologists are apathetic towards systems they deem "impersonal."

This paper investigates efforts between 1959 and 1968 by academics, IBMers, and mainframe customers to make IBM mainframes more personal through the use of timesharing

[1] A recent photograph revealed that one of the earliest users of the SAGE used the light pen to draw a "sexy dame." This was possibly done by an IBMer responsible for training of young Air Force SAGE operators (Edwards 2013).

and virtualization. It argues that the breakthroughs pioneered by MIT's Compatible Time-Sharing System (CTSS) and the IBM Cambridge Scientific Center's CP/CMS revolutionized computing by defrocking data processing middlemen and giving end-users direct computer access via human-friendly single user interfaces. Additionally, CP/CMS's invention of virtualization extended this personal interaction from small "extended machine" sandboxes to virtual machines capable of running complete operating systems, such as CMS and OS/360. In contrast to efforts by IBM marketing to make mainframes likeable to executives,[2] CP/CMS made mainframes likeable to end users by re-forging Big Iron into tools for individual productivity. Given these revolutionary aspects, these innovations proved to be quite controversial, pitting end-users against data processing professionals, academia against industry, hardware against software, and traditional IBM conservatism against Cambridge hacker dynamism.

This story is worth telling for a number of reasons. First, it exposes that the period from 1959 to 1968 was exceptionally seminal for the development of personal computing, suggesting that the PC "Revolution" was actually part of a larger evolutionary process. Second, CP/CMS pioneered virtualization, a technology that is now central to most every form of computing. This heritage is of particular interest to IT professionals that may not realize that virtualization long predates the 1998 formation of VMWare. Third, computer histories currently ignore or downplay the technological significance of CP/CMS, particularly in its role as the "missing link" in the evolution of timesharing supervisors to virtualization hypervisors. Finally, the story of CP/CMS shows how a small group of intelligent and resourceful IBM "Wild Ducks"[3]

---

[2] IBM Marketing has occasionally sought to make the purchase of a mainframe evoke in client executives the excitement of buying a new sports car. This is illustrated by the availability of the System 360 in your choice of colors (Willow Green, Garnet Rose, Sunrise Yellow, Classic Blue, Charcoal Brown, or Pebble Grey) or designing the case of the modern zEC12 mainframe to suggest the mystique of the F-117 Stealth Fighter.

[3] ***wild duck*** n. A creative technical person who does unconventional things, or at least does things in an unconventional way. Implies respect, and an acknowledgement that many of that person's ideas turn out to be valuable. It is said that IBM does not mind having a few wild ducks around − so long as they fly in formation.

successfully developed, protected, and promoted a skunk works project that ran counter to IBM culture and strategy to provide a personal, productive, and extensible computing experience.

Batch Processing: Industrial-Strength Computing

Would you rather fill a cookie jar by baking one cookie at a time or by baking multiple cookies together in a single batch? Although one could certainly preheat the oven, measure out the ingredients for a single cookie, beat one egg, mix the dough, roll it into a single ball, and stick it into the oven, chances are that you would pursue the batch approach to avoid wasting time. After all, if you are going to go through the hassle of each step of your grandma's famed cookie recipe, you might as well get more than one cookie out of the effort! This idea is based around the principle of *economies of scale*, which dictates that when the marginal cost of baking an additional cookie is lower than that of baking the previous cookie, the average per-cookie cost drops by increasing the size of the batch. During the dawn of industrial mass production, smart managers, such as Henry Ford, realized that maximizing profit closely related to maximizing the utilization of tools and labor. This lead to the concept of *minimum efficient scale*, below which it was unprofitable to bake a cookie, manufacture a car, or assemble a widget. In the early days of computing, these same economic forces affected the manner in which companies ran their computing workloads. Because computers were expensive centralized resources that could only do one thing at a time and required significant time to be reprogrammed to handle different problems, companies based their data processing operations around this concept of minimum efficient scale. This led to a system of *batch processing*, whereby companies maximized utilization by grouping similar workloads together into large batches to minimize the frequency and duration of computer downtime for reprogramming.

This term was created by T. J. Watson Jr., who told a story (by the philosopher Kirkegaard) about a flock of wild ducks that landed near a farm. Some got fed by the farmer and stayed, and either died of obesity or got eaten. The truly wild ones flew away − and survived. (Colishaw 1990)

Just as the practice of large-scale manufacturing depended on a specialized group of managers, batch processing depended on a new group of specialists called data processing professionals. Drawing from disciplines such as scientific and operations management, data processing professionals engineered batch processing workflows to maximize throughput of batch jobs. Due to the technical specialization of these individuals and the fact that computers were too costly for individual departments, data processing was typically centralized in a distinct department. The responsibilities of this department included establishing batch schedules, queuing requests by individual departments, assigning batch windows, running jobs, printing and delivering reports, running a system of charge-back based on computer usage, and maintaining computer equipment. This process meant that if an end user wanted to use a computer, they would have to follow a defined procedure:

1. *Encode the job on punched cards or magnetic tape using a keypunch or small computer.*
2. *Request resources from the data processing department.*
3. *Wait for the allotted batch window.*
4. *Deliver the punched cards or magnetic tape containing the job to the data processing department in person or via courier to be run on a larger computer.*
5. *Wait hours or days to receive the print out of results.*
6. *Read and process the results.*
7. *Determine if a miscalculation or programming error has caused the job to abnormally end (called an ABEND). If so, correct the issue and restart the procedure at step 1.*

Due to these methods, batch processing proved most effective for workloads, such as accounting, quarterly reports, or payroll, which depended on static and repeatable business processes. However, scientific and engineering customers, such as universities or labs, often lacked these sorts of repeatable business processes. Their end-users handcrafted high-level programs in FORTRAN, MAD, BASIC or LISP that would only be run once or a handful or times, leading to workloads inherently dynamic, error prone, and experimental. In this environment, batch processing proved a suboptimal, frustrating, and fatiguing system that forced end-users to iterate

through several batch jobs to get a single program to execute correctly. After all, can you imagine waiting days or weeks between submitting your program and receiving a printout of your results, only to find out that a typo in your FORTRAN program ABENDed the job, forcing you to reschedule and re-run the program? For these reasons, scientific and engineering end-users considered batch processing a "clumsy" interface that created "stifling orderliness" through bureaucracy.[4] These frustrations ultimately proved a powerful motivation for MIT students and professors to innovate the human-computer interface.

The Timesharing Revolution

On May 9, 1963, a television program called the MIT Science Reporter gave the viewing public a peek at timesharing technology (MIT Science Reporter, 1963). Visiting the MIT Computation Center, the reporter John Fitch interviewed MIT Professor Fernando J. Corbató regarding his work to develop the *Compatible Time-Sharing System* (CTSS). According to Corbató, most of the problems surrounding computing hardware reliability and programming language efficacy had been solved in the 1950s, making the core challenge of the 1960s improving the end-user interface. Given that a computer was limited to sequentially solving one problem at a time at a cost between $300 and $600 per hour ($2,250 and $4,500 in 2013 USD), Corbató opined that MIT was other organizations were initially forced to adopt a system of batch processing. However, this system was "pretty clumsy because of the weak interaction between

[4] Hacker mythology uses the IBM of the bath-processing era as a foil for the hacker identity. *"The epitome of the bureaucratic world was… a very large company called… IBM. The reason its computers were batch-processed Hulking Giants was only partially because of vacuum tube technology. The real reason was that IBM was a clumsy, hulking company that did not understand the hacking impulse. If IBM had its way…, the world would be batch processed, laid out on those annoying little punched cards, and only the most privileged of priests would be permitted to interact with the computer. All you had to do was look at someone in the IBM world and note the button-down white shirt, the neatly pinned black tie, the hair carefully held in place, and the tray in punched cards in hand. You could wander into the Computation Center, where the 704, the 709, and later the 7090 were stored – the best IBM had to offer – and see the stifling orderliness, down to the roped-off areas beyond which unauthorized people could not venture… IBM [was] an empire unto itself, secretive and smug… They were batch-processed people, and it showed not only in their preference of machines, but in their ideas about the way a computation center, and a world, should be run."* (Levey, 2010, 41)

the man and the machine," leading MIT to rethink their computer environment to provide a more responsive end-user interface. Rather than queuing and running each batch job from start to finish, computing resources could be broken into bursts that could be allocated to multiple programs in a round-robin fashion called *multiprogramming*. Running a computer would be less like a marathon, where each program monopolizes the computer for a prolonged period, and more like a relay-race, where each program sprints for a short duration before "passing the baton" to another program, even if this means that each program requires several such sprints to complete. By connecting each program in this relay race to typewriter terminals, computers can thus be timeshared to allow multiple users simultaneous real-time interaction. As improbable as this all likely sounded to viewers, Corbató proved the concept by demonstrating how a novice end-user could easily use a terminal connected to CTSS to calculate either the area of a square or the hypotenuse of a triangle.

The road to Corbató's 1963 CTSS demonstration began in the late 1950s when MIT Professor John McCarthy theorized that computers and teletypes could provide "an operating system that permits each user… to behave as though he were in sole control of a computer, not necessarily identical with the machine on which the operating system is running" (McCarthy 1983). While numerous minds at that time were thinking about using multiprogramming to run several batch jobs simultaneously, McCarthy appears to have been the earliest figure to envision the use of multiprogramming and teletypes to provide a more personal computer interface. Most likely, these views evolved out of McCarthy's theoretical work in artificial intelligence, which included combining an IBM 709 mainframe with a Flexowriter teletype to allow a human to play chess against a computer opponent. Additionally, McCarthy and other MIT faculty were exposed to real-time computing relatively early through the MIT Lincoln Lab's involvement in the

Whirlwind project and the subsequent formation of the Digital Equipment Corporation (DEC) by Lincoln Lab employees to sell low cost interactive minicomputers (Levey 2010, 26-94).[5] By January 1, 1959, these experiences convinced McCarthy that real-time interactive computing was the wave of the future, leading him to draft a memo proposing that MIT work to timeshare their IBM 709.

By the time that Fernando Corbató began to design and build CTSS, MIT had benefitted from several free upgrades to their IBM-donated 709, ultimately giving them the same $3,000,000 IBM 7090 used by NASA on the Gemini and Saturn projects.[6] While the 7090 was impressive, McCarthy and his peers understood that a timesharing system would be more memory-intensive than the sorts of processor-intensive workloads run by NASA, leading them to solicit IBM for a second IBM 7302 Core Storage Unit (Walden and Van Vleck 2011, 3). Once upgraded, this provided MIT a system with two separate memory banks (each approximately 72 kilobytes), which reflected McCarthy and Corbató's decision to functionally divide the CTSS architecture between a supervisor module and multiple extended machines for each attached end-

[5] The MIT Lincoln Lab donated the diminutive TX-0 computer to MIT in 1958 and DEC donated the PDP-1 in 1961. In comparison to the donated IBM machines, these smaller machines were more interactive and free from the control of the Data Processing department. Given that these computers were designed to allow a user to write programs directly to paper tape and interact with them in real-time at the console, these systems enabled the "hacker's nirvana" by providing the MIT students of the Tech Model Railroad Club (TMRC) the hitherto unheard of chance to directly "hack" at a computer. Considering that computer time was worth the equivalent of thousands of dollars per hour, this was an incredible opportunity. The TMRC students created numerous notable programs, including a real-time debugger called FLIT, a primitive program to produce electronic music, and a series of computer games, including Spacewar!

[6] In 1955, IBM President Tom Watson Jr. decided that the fact that "there was not a single university with a computer curriculum" hurt computer sales because customers were worried about "hiring somebody who could run it." IBM had attempted to partner with Harvard during the 1940s, but fallout with Dr. Howard Aiken over the IBM Automatic Sequence Controlled Calculator (a.k.a. the Harvard Mark I) made further cooperation impossible. Tom Watson Jr. thus gave MIT an IBM 709 and a staff of IBMers to run it to urge them "to start training computer scientists." Over twenty years, IBM donated computing hardware and expertise worth hundreds of millions of dollars to help MIT become the leader of the embryonic field of computer science. This largesse ensured that all nearly all of IBM's early research was conducted on IBM hardware. (Watson 1990, 244-245)

user (Fano 1964).[7] Under this architecture, the CTSS supervisor would automate and abstract away most aspects of running the computer hardware, including process management, memory management, I/O, file management, command processing, and device management. Each end-user would be provisioned a dedicated address space of virtual memory called an extended machine to facilitate the illusion of each user having a dedicated general-purpose computer. Rather than having to manually deal with things like I/O channels, each extended machine would use an application-programming interface (API) to request computing resources from the supervisor via supervisor calls (SVCs) (Saltzer 1963). CTSS thus pioneered the division between kernel and userland used by all modern operating systems.

The CTSS project ultimately proved to be a resounding success for MIT and its pioneering professors. While most computer manufacturers were conservatively focused on the traditional batch workloads, MIT's CTSS positioned academic computer scientists in the vanguard of software innovation around improving the end-user interface.[8] CTSS leveraged the early lessons of the SAGE project, popularized the concept of a general-purpose timesharing system, and equipped MIT to win a two million dollar DARPA grant for ongoing research around the man-machine interface. As a result of this grant, MIT founded a new center called Project MAC (Mathematics and Computing), and committed to furthering the innovations of CTSS through the creation of a next-generation timesharing system called Multics (Multiplexed Information and Computing Service).

---

[7] This architecture reflected an evolution of monitor programs, such as the FORTRAN Monitoring System (FMS), which pioneered the separation of systems programs dedicated to managing the computer from the application programs dedicated to accomplishing some unit of work.

[8] This assessment is most true in the area of general-purpose timesharing systems, which provided users access to the full facilities of the computer. IBM was actually a key innovator in earlier and more restrictive forms of timesharing, including the timesharing of a particular application via the SAGE and SABRE systems and the timesharing of a particular programming language environment via the QUIKTRAN system. However, none of these systems had the scope of vision pioneered by CTSS.

When MIT released a Request for Proposal (RFP) for a computer to run Multics, the Cambridge IBMers lobbied headquarters for dedicated resources to write a winning proposal.[9] In response, IBM headquarters sent Norm Rasmussen to expand the existing MIT Liaison Office into a full-fledged IBM Cambridge Scientific Center (CSC) with the dual mandate of managing the MIT relationship and ensuring IBM leadership of the field of computer science. In order to maximize collaboration with Project MAC, Norm opened the CSC in February 1964 on the 4th floor of 545 Technology Square, a mere one floor below Project MAC.[10] Over the following months, Norm got settled and began meeting the major players in the Multics project. This situation changed dramatically on April 1, when IBM announced the System/360 family of computers. As the very first compatible family of computers ranging from very small to very large that could handle both scientific and business applications, this announcement was positive for most organizations.[11] However, it proved a major disappointment to the Project MAC leaders due to System/360's lack of support for dynamic address translation (DAT), a technology

[9] While the IBM corporate leadership had a lukewarm attitude towards timesharing, the IBM team in Cambridge understood and supported MIT's timesharing research. Under the terms of T.J. Watson Jr.'s 1955 donation of the IBM 709, the IBMers of the MIT Liaison Office helped operate and maintain MIT's IBM equipment, leading them to work hand-in-glove with researchers such as Professors John McCarthy and Fernando Corbató. This gave them insight into real-time computing and the potential that smaller and nimbler companies like Digital Equipment Corporation (DEC) had to erode IBM's market leadership.

[10] Around the same time that IBM opened the Cambridge Scientific Center on the 4th floor, General Electric opened up their Cambridge Information Systems Laboratory on the 7th floor. For more information, see Tom Van Vleck's description of the building at http://www.multicians.org/tech-square.html.

[11] "The decision by the management of [IBM] to produce… the System/360 has emerged as the most crucial and portentous— as well as perhaps the riskiest—business judgment of recent times. The decision committed I.B.M. to laying… some $5 billion over a period of four years… Not even the Manhattan Project, which produced the atomic bomb in World War II, cost so much (the government's costs up to Hiroshima are reckoned at $2 billion), nor, probably, has any other privately financed commercial project in history… To launch the 360, I.B.M. has been forced into sweeping organizational changes, with executives rising and falling with the changing tides of the battle. Although the fact has largely escaped notice, the very character of this large and influential company has been significantly altered by the ordeal of the 360, and the way it thinks about itself has changed, too... The new System/360 was intended to obsolete virtually all other existing computers—including those being offered by I.B.M. itself. Thus the first and most extraordinary point to note about this decision is that it involved a challenge to the marketing structure of the computer industry—an industry that the challenger itself had dominated overwhelmingly for nearly a decade. It was roughly as though General Motors had decided to scrap its… makes and models and offer… one new line of cars, covering the entire spectrum of demand, with a radically redesigned engine and an exotic fuel… Bob Evans, the line manager who had the major responsibility for designing this gamble of a corporate lifetime, was only half joking when he said: "We called this project `You bet your company.'" (Wise 1966).

particularly critical for timesharing systems (Parmelee, et al. 1972).[12] This realization possibly confirmed to Fernando Corbató that IBM cared little for timesharing innovation, leading him to criticize the System/360 architecture at a meeting of the SHARE user group (Varian 1997). Over the subsequent months, Norm Rasmussen attempted to move heaven and earth to win the Multics RFP, including bringing the System/360 Chief Architect Gene Amdahl to MIT, but Project MAC ultimately rejected IBM's proposal to provide a System/360 modified to use DAT via a "Blauuw Box." Instead, they awarded the Multics contract to General Electric for their GE-645, an IBM 7090 compatible that featured out-of-box support for dynamic addressing.

CP/CMS and the Realization of Virtualization

When CTSS was finally shut down for good on July 20, 1973, Peter Roach, the primary CTSS administrator "felt sad," but he was consoled by the fact that he was "pretty involved" in two new systems which he considered "the next evolution" of CTSS technology (Walden and Van Vleck 2011). The first system was Multics, the timesharing system designed by Project MAC as the successor to CTSS. The second was an IBM timesharing system called CP/CMS. This consolation was striking for two reasons. First, it demonstrated that an MIT employee with deep knowledge of CTSS and Multics considered CP/CMS to be a co-heir of the CTSS legacy. Second, it reflected the fact that an MIT department chose to run CP/CMS over Multics, leading an MIT systems administrator to support a production CP/CMS environment. Considering that the Multics effort was led by a "jelled team" of CTSS veterans under Fernando Corbató, this

[12] IBM made the conservative architectural choice to avoid virtual memory in the System/360 because of concerns about "thrashing" that had plagued the Ferranti Atlas and worries that virtual memory may not have been possible on the smaller members of the System/360 family. However, it seems that the high-end IBM 8000 series machines designed to replace the 7090s that ran CTSS did actually plan to implement virtual memory. This is why Gerrit Blaauw, who had worked on the 8000 series, designed a "Blauuw" box to enable virtual memory on the System/360. Ironically, IBM Vice President Vin Learson's decision to cancel the 8000 series and replace it with the System/360 family was influenced by a 1961 trip to MIT. Based on the timing, it is possible that he saw MIT's recently installed PDP-1 minicomputer.

outcome would have been surprising to a casual observer in 1964 (Walden and Van Vleck 2011). If a "clumsy, hulking company" like IBM that sold "batch-processed Hulking Giants" failed to win the Project MAC RFP or provide dynamic addressing in their System/360 architecture, how could they possibly challenge MIT leadership in timesharing?

IBM's loss of Project MAC to General Electric was a particular painful blow. IBM had invested heavily in MIT to help them become a leader in the field of computer science, so the symbolic repercussions of this snub were severe. While the System/360 was setting unprecedented sales records, many IBMers were getting the impression that the timesharing situation was spiraling out of IBM's control, making their lack of a viable solution a strategic threat to the new architecture (Von Vleck 2010). Indeed, as if by magic, Dartmouth, UC Berkeley, and the RAND Corporation announced new timesharing systems, all of which did not run on IBM hardware (Walden and Van Vleck 2011). In this dark cloud, Norm Rasmussen continued to build CSC and keep tabs on Project MAC one floor up. At some point, he decided that, since Multics would not run on System/360, IBM should build their own timesharing system. Additionally, since the success of CTSS had effectively turned the Tech Square building into a tech incubator for timesharing, the CSC would be well equipped for this project. Considering that the CSC had already attracted Lyndalee Korn, one of the seven authors of the original CTSS paper, Norm began recruiting other CTSS veterans to "come downstairs" and join the IBM team on the 4th floor (Varian 1997).

The most important CTSS veteran to "go downstairs" was Bob Creasy, a software developer for the CTSS supervisor that became dissatisfied with Project MAC for two reasons. First, he believed that Project MAC was pursuing a poor architectural plan for Multics, as the

interface between the supervisor and the extended machines had made CTSS inflexible.[13] Rather than using SVCs, he believed that the extended machine interface should be made identical to the basic machine interface through software emulation. Second, he believed that the senior Project MAC leaders had made a mistake in awarding the contact to General Electric. While he agreed that dynamic addressing was critical to timesharing, he considered this to be minor compared to the innovations of the System/360, whose Principles of Operations documentation could serve as a blueprint for building the first hypervisor.[14] For these reasons, Bob Creasy accepted Norm Rasmussen's offer of leadership of the IBM timesharing effort, and "went downstairs" to implement his vision for a Virtual-Machine Timesharing System.

While Norm Rasmussen and Bob Creasy began planning their timesharing project, another part of IBM discovered a possible reprieve from the MIT fiasco. The University of Michigan had long been a flagship IBM campus, second only to the MIT Computation Center. After MIT's snubbing of IBM for Project MAC, Professors Bernard Galler and Bruce Arden sensed an opportunity to overcome IBM's traditional predilection for elite East Coast universities and position the University of Michigan as IBM's new flagship campus (Akera 2008). Contacting IBM and the National Science Foundation, these professors suggested that they could build a System/360 timesharing system, as long as IBM followed their specifications for hardware changes to enable dynamic addressing. IBM tentatively accepted to this proposal, but soon changed course after reaching out to customers to help the professors gather a list of requirements. Because customer demand for timesharing far exceeded their expectations, IBM decided to instead develop the timesharing system internally as an official supported product

[13] "Although successful as a production system, the interconnections and dependencies of its supervisor design made extension and change difficult" (Creasy 1981).
[14] "The family concept of the IBM System/360… was a most amazing turning point in computer development… We believed that the architecture… would be around for a significant amount of time." (Creasy 1981)

released alongside a new System/360 model with DAT capabilities. However, the University of Michigan and five other clients under non-disclosure agreements would form the "Inner Six" and help IBM steer the direction of the product via SHARE user group committee (Varian 1997).

At some point in this process, IBM headquarters invited IBM timesharing experts, including Norm Rasmussen, Bob Creasy, and other CSC members, to a planning task force for this effort. However, this was likely after IBM and the 'Inner Six' had established a high-level architectural plan for the future timesharing solution. At most, the task force likely sought practical recommendations about logistics and project management, and perhaps suggestions about whom should lead the project. Under this situation, there was little leeway for Bob Creasy to convince the committee of the merits of a virtual machine approach to timesharing or influence the architecture or development plan in a meaningful way.[15] This is reflected in the task force's decisions to make TSS/360 a supervisor-based system like CTSS and Multics, have development effort led by Andy Kinslow, an IBMer that had developed a CTSS-clone for the 7090 called the "Big Old Supervisory System," and use New York-based IBM developers coming off the OS/360 project. Shortly after this task force, IBM officially announced Time Sharing System/360 (TSS/360) and System/360 Model 67, a system featuring DAT capabilities based on the specifications of the University of Michigan professors, in August 1965 with a targeted release of July 1966 (Pugh, Johnson and Palmer 1991, 362).

The realization that IBM committed to the same architecture as Project MAC and that the CSC personnel would have no role in IBM's strategic timesharing solution must have been a

[15] This is possibly due to several reasons. First, the loss of Project MAC may have hurt the credibility of the IBM Cambridge Scientific Center. Second, at 25, Bob Creasy was likely far younger than the rest of the committee, which may have led the older IBMers to discount this young MIT graduate. Third, IBM reluctance to adopt virtual memory in the System/360 reflected a concern about excessive virtualization overhead hurting performance. Now that they were not considering adding virtual memory, the idea of virtualization may have seemed far too risky for a product that needed to come to market ASAP to resort IBM credibility.

considerable shock to Bob Creasy. He joined IBM in the understanding that he would use CSC resources to develop his virtual-machine timesharing concept, but given IBM's decision to pursue TSS/360, any timesharing project at CSC now risked the danger of being labeled "counter-strategic" to IBM. Considering that IBM Vice President Vin "the Hatchet Man" Learson had recently unified IBM's hardware product line through the brutal suppression of competing factions, a CSC timesharing system seemed a risky proposition. On the other hand, the CSC members thought that TSS/360 would be unsuccessful due to the reports of OS/360 performance and compatibility issues, and the fact that the IBM developers coming off OS/360 lacked experience with interactive computing (Brooks 1995). In contrast, the CSC had experience developing a timesharing system, a superior architecture based on virtualization, and a ready pool of timesharing experts to draw from should CSC need to expand. Due to these sentiments, Norm asked Bob in the fall of 1964 to move forward with their original plan to develop his virtual-machine timesharing design. Bob took some time to contemplate the decision, and then "launched the effort between Xmas 1964 and year's end, after making the decision while on an MTA bus from Arlington to Cambridge" on "a Tuesday" (Creasy 1981).

In the last week of 1964, Bob Creasy and a CSC coworker named Les Comeau began planning work for the CSC timesharing system. They started by detailing the plan for creating the *Virtual Machine Command Program* (CP), a hypervisor that would run on a System/360 in a privileged state and simulate virtual System/360s using a software implementation of the System/360 Principles of Operation. However, they soon realized that their hypervisor would be insufficient for a complete time-sharing solution, as the existing System/360 operating systems were all either too batch-oriented or too complex to serve as an effective end-user operating system. This led to the formation of a second team to create the *Cambridge Monitor System*

(CMS), which would be a simple single-user operating system based closely on CTSS.[16] As CP and CMS together formed a timesharing solution, the CSC team began to refer to their system as CP/CMS. Finally, Creasy and Comeau realized that it would be a while before they could acquire a 360/67 model, as they were not part of the 'Inner Six' or TSS/360 development. In order to expedite CP/CMS development, they decided to add dynamic addressing to their 360/40 by building a Cambridge Address Translator (CAT) box. Work on these three components proceeded rapidly when the rest of the Cambridge team returned in early January. Within a month, the CSC team released their first internal IBM report on CP/CMS, seeking to allay fears of counter-strategic behavior by characterizing it as research tool for studying the performance of interactive workloads on virtual memory systems. However, it seems that the TSS/360 team nonetheless expressed discontent with the idea that Bob Creasy decided to move forward with the development of his ideas for a virtual-machine timesharing system. Reflecting back on this time period, Bob Creasy stated, "in January 1965… it became apparent… that the system would be controversial" (Creasy 1981). In spite of this controversy, or perhaps because of it, Bob Creasy drove the CP and CMS teams hard and covertly increased CSC headcount by hiring MIT students on a part-time basis. By mid 1965, the designs for CP and CMS were complete, and the CSC team began to implement their code.

At some point in early 1965, Norm Rasmussen and Bob Creasy faced one of his greatest fears: a surprise inspection of the CSC by IBM Vice President Vin Learson. As the corporate "Hatchet Man" known for inflicting pain to ensure the IBMers tow the company line, Learson personified the terrible consequences that could befall the CSC should IBM sense that CP/CMS

[16] CMS was therefore the first "personal computer" operating system. At the time, Bob Creasy saw CMS as a means to give the CSC staff an enjoyable interface along the lines of CTSS. Given that the "Control Program… give[s] each user the choice of any software, whether backlevel, standard, modified, or home-brewed" (Creasy 1981), Bob "didn't expect the design to hold for more than a couple of years," as users would be free to select from any number of operating systems (Varian 1997).

was a threat to the success of TSS/360. Luckily, the visit seemed to have been relatively harmless. Rasmussen did panic because he was out of IBM uniform ("Oh, my God, I'm wearing a blue shirt"), but the young Lyndalee Korn successfully entertained Learson by demonstrating some of the CTSS features she developed before coming to IBM and explaining that the CSC planned to develop a similar program for the System/360 (Varian 1997). Based on some reports that Learson had a fondness for female IBMers (his secretary supposedly nicknamed him "Pinchy" for the way he treated her in the elevator), perhaps it was advantageous that Lyndalee was on hand to explain CSC to him (Bemer n.d.). Indeed, as a result of this visit, Vin Learson sent the CSC as spare System/360 while their primary system was down for being upgraded with their CAT box.

In August 1966, the TSS development team informed the 'Inner Six' that it was unlikely that the Model 67 would be released before December or that TSS would be released before April 1967. Even worse, the first release of TSS would only be for "experimental, developmental, or instructional use" (Pugh, Johnson and Palmer 1991, 362). The reasons for this were myriad. According to a member of the TSS architecture group, "OS/360 hadn't settled town sufficiently when TSS began, and there was too much of a rush to completion" (Goodwin 2009). Out of the 'Inner Six,' the University of Michigan was especially frustrated by these delays, as they had promised that that timesharing services would be available by the fall semester of 1966. Professor Bernard Galler accused IBM of "attempt[ing] from the beginning to build a system and included everything except the kitchen sink" (Akera 2008). Having obediently cast aside their aspirations to develop a timesharing system in-house to support TSS two years earlier, the Michigan Computer Center now believed that they had misplaced their faith in the abilities of IBM development, leading them to resume development of their own *Michigan Terminal System*

for the System/360.[17] These adverse announcements rocked the company similar to the loss of Project MAC, leading IBM CEO Tom Watson Jr. to admit in the 1966 Annual Report that IBM had "experienced delays in meeting our original objectives for… time-sharing systems" (Pugh, Johnson and Palmer 1991, 362).

By January 1967, Bob Creasy and the rest of the CSC team were in a celebratory mood. Despite the delays and performance issues experienced by TSS and Multics, CP and CMS were now running production workloads on a regular user schedule. Even OS/360 was successfully running as a CP guest operating system, which gave the CSC team a glimpse into some of the powerful possibilities of virtualization (Goldberg, Architectural Principles for Virtual Computer Systems 1973). By using the virtual cardpunch and readers associated with each virtual machine, guest operating systems could actually communicate with one another by writing and reading virtual punch cards to and from a shared CP spool area (Hendricks and Hartmann 1979). This allowed end users to write programs in the user-friendly CMS, submit them to the job entry subsystem of OS/360 for batch processing, and then receive and view the job output in CMS. Combining CP/CMS with OS/360 therefore allowed end-users direct access to batch processing at memory speed without a data processing middleman. Additionally, virtual machines proved a powerful environment for operating system and subsystem development that traditionally would require downtime for an entire mainframe. Rather than having to code in the middle of the night to avoid disrupting production workloads, virtual machines gave systems programmers more flexibility and sanity in their day-to-day work.

Over the next year, the technical superiority of CP/CMS and the negative fallout around TSS/360 caused dramatic reversals in IBM timesharing systems. In early 1967, the 'Inner Six'

[17] This system became the Michigan Terminal System, which was operational six months after delivery of their Model 67, fully developed by late 1968, and run across a consortium of several universities by 1969. (Akera 2008).

finally received a copy of TSS for their new System/360 Model 67s, which led them to observe the reported performance and stability problems firsthand. A systems programmer (administrator) noted that the system "took a long time to IPL (boot) and did not stay up long" (Varian 1997, 17). This led to additional members of the 'Inner Six' to follow the University of Michigan's lead and defect from the TSS project. Due to their proximity to the CSC, the staff at the MIT Lincoln Lab knew firsthand of CP/CMS, leading them to fully replace TSS with CP/CMS by April and demand that IBM offer formal support. Similarly, Union Carbide demanded CP/CMS in place of TSS once a vice president heard about the system from an IBMer living next door. To show their conviction, Union Carbide even sent two of their programmers on loan to the CSC to help Bob Creasy add new functionality. This second and third high-profile defection from the 'Inner Six' ultimately served as coffin nails for TSS/360. While the TSS development team began scheming with some IBM Research staff to find a way to suppress and destroy CP/CMS, this proved impossible. The CP hypervisor seemed irresistible to many researchers and developers, and by September 1967, a member of IBM Research drove from Yorktown Heights to Cambridge biweekly to pick up the latest version of CP/CMS. By the spring 1968 meeting of SHARE, IBM de-committed TSS[18] and the CSC gave a presentation on CP/CMS to the TSS user group. In June 1968, CP/CMS became formally available at an IBM Type III Program.

[18] TSS has become the textbook case of software development failure. Reflecting back, the TSS product manager, Andy Kinslow, stated "anyone who's been on one large failure will make sure that they are never on another. Therefore, most large projects are staffed with trainees and masochists." (Orr 2004)

Conclusion

The ten years from 1959 to 1968 were some of the most innovative in the history of computing. In 1959, all computers were shared infrastructure run through batch processing administered by data processing departments, and timesharing was a mere idea in the head of a forward-thinking MIT professor. In 1968, increasing numbers of end-users had direct access to computers through the use of teletypes or display terminals with timesharing operating systems. In the case of CP/CMS, end-users literally had complete access to their own virtual personal computer running a single-user operating system on the System/360 architecture. As this narrative demonstrates, innovation between these two points followed a crooked path with numerous forks in the road and dead-ends. Progress required visionaries like John McCarthy and Fernando Corbató to discern a vision for the future and wild ducks like Bob Creasy and the rest of the CSC team to challenge the status quo in spite of discomfort and career risk. By envisioning and fighting to make computing accessible, user-friendly, and personal, the pioneers of this decade unleashed forces of creative devolution similar to the printing press, setting the stage for personal computing.

While most of the traditional timesharing systems would eventually wiped out by the 1980s by hardware commoditization and desktop computers, the software innovations of this period continue to cast a long shadow on modern IT. Due to its role as the first hypervisor, CP has survived by evolving into z/VM, IBM's premier solution for large-scale consolidation of virtualized Linux workloads. It has since influenced the development of x86 hypervisors, including VMWare, Xen, and KVM, and other implementations of virtualization, such as the Java or Dalvik Virtual Machines (Rosenblum 2004). Although Multics had limited commercial success due to performance issues, both it and CTSS significantly influenced the design of the UNIX family of operating systems, including Linux (Ritchie and Thompson 1974).

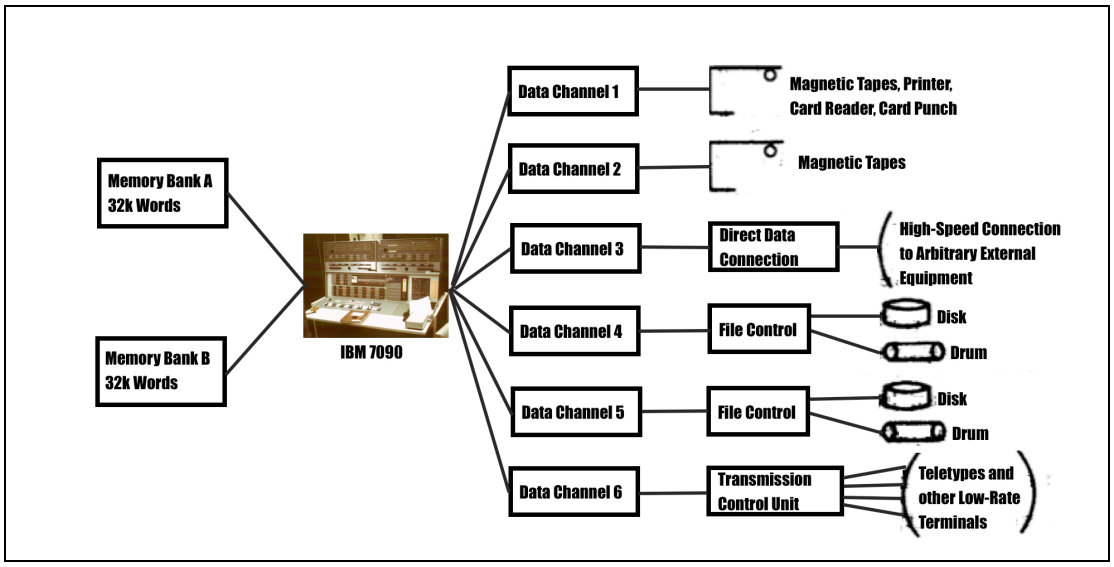

Figure 1: Physical Architecture of MIT CTSS

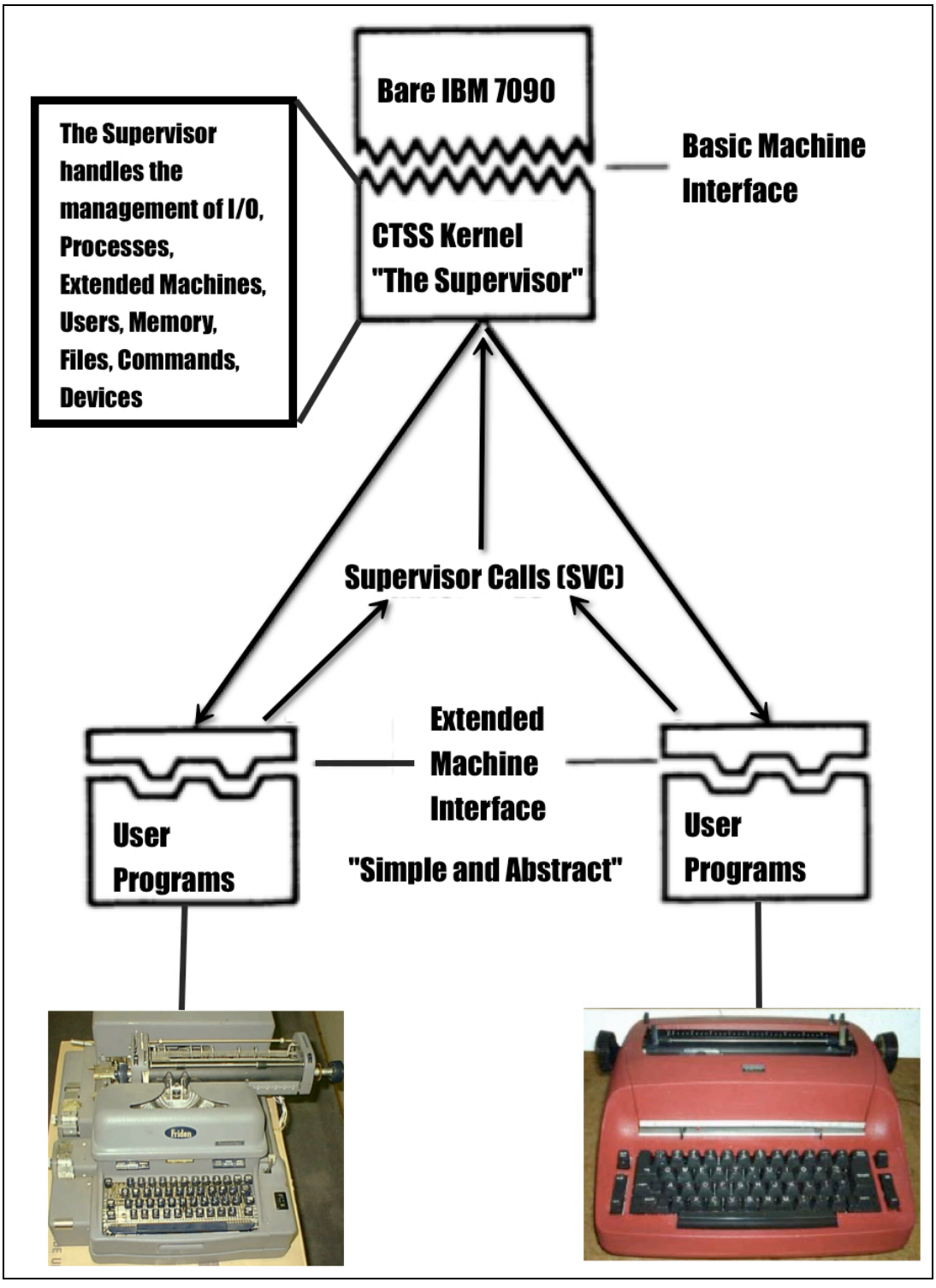

Figure 2: Logical Architecture of MIT CTSS

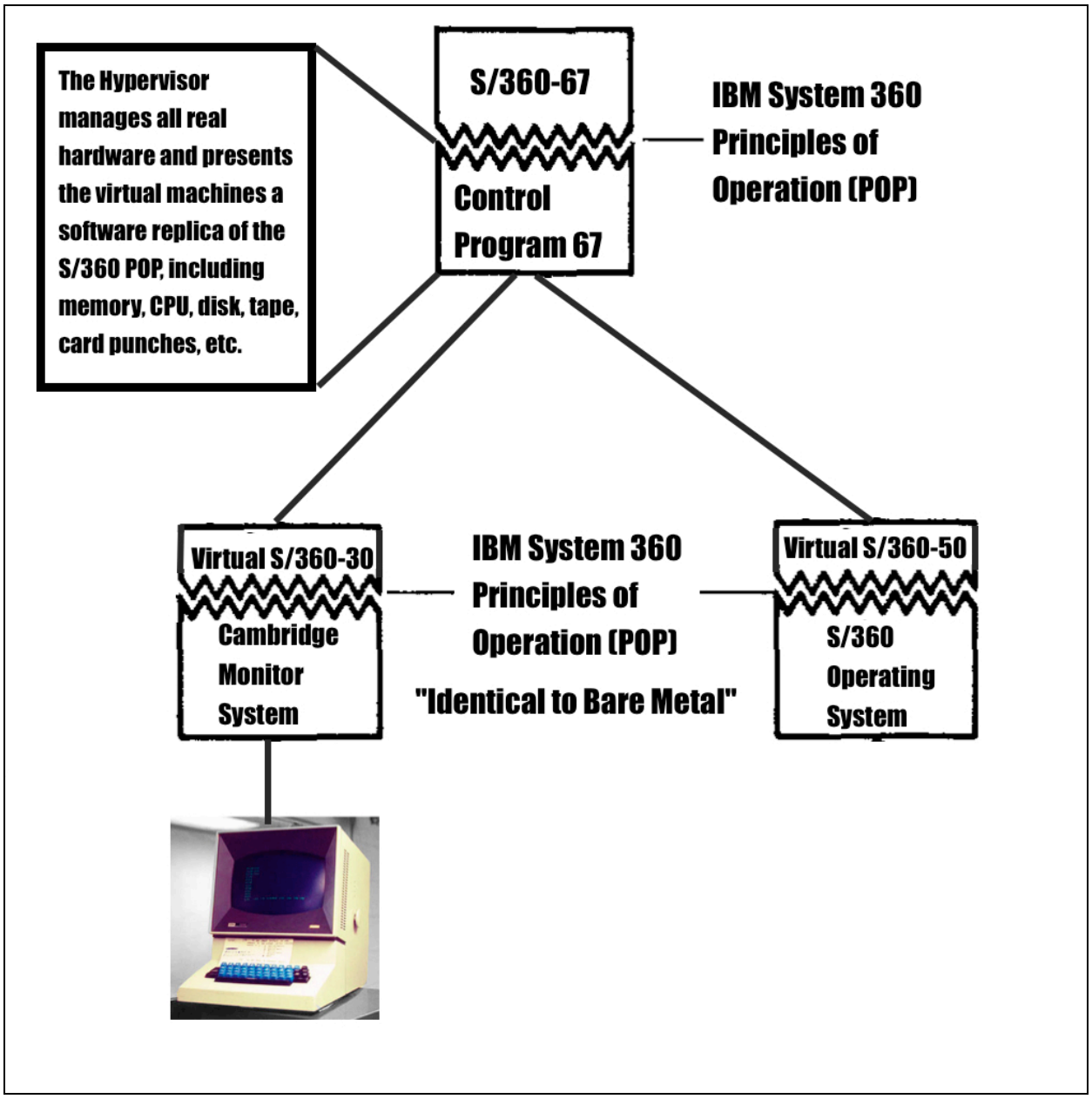

Figure 3: Logical Architecture of CP/CMS

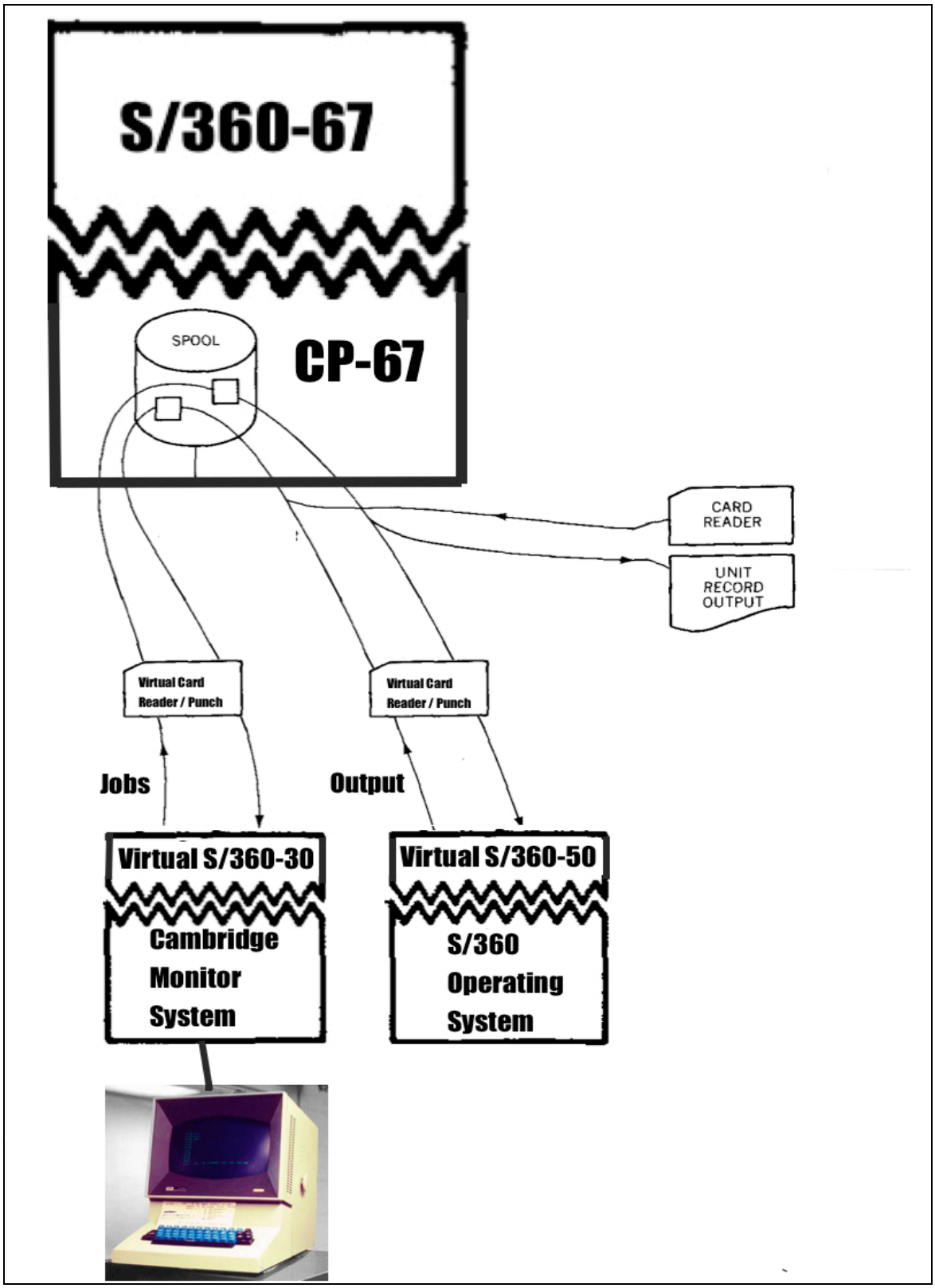

Figure 4: Logical Architecture of Communication between CP/CMS Guests via of SPOOL Area

Works Cited

Akera, Atsushi. "The Life and Work of Bernard A. Galler (1928 - 2006) ." *IEEE Annals of the History of Computing* (IEEE Computer Society) 30 (2008): 4-14.

Bemer, Bob. "IBM - Why I left for UNIVAC." *Thoughts on the Past and Future with computer pioneer Bob Bemer.* http://www.bobbemer.com/LEAVEIBM.HTM (accessed March 30, 2013).

Brooks, Frederick Jr. *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition.* 2nd Edition. Boston: Addison-Wesley Pub., 1995.

Computer History Museum. "Bytes for Bites: The Kitchen Computer." *Computer History Museum.* http://www.computerhistory.org/revolution/minicomputers/11/362 (accessed March 30, 2013).

Creasy, R. J. "The Origin of the VM/370 Time-Sharing System." *IBM Journal of Research and Development* (IBM Corporation) 25, no. 5 (1981): 483-490.

Edwards, Benj. "The Never-Before-Told Story of the World's First Computer Art (It's a Sexy Dame)." *The Atlantic.* January 24, 2013. http://www.theatlantic.com/technology/archive/2013/01/the-never-before-told-story-of-the-worlds-first-computer-art-its-a-sexy-dame/267439/ (accessed March 30, 2013).

Fano, R. M. *The MAC System: A Progress Report*. Cambridge: MIT Press, 1964.

Goldberg, Robert P. *Architectural Principles for Virtual Computer Systems.* Cambridge, MA: Harvard University, 1973.

Goodwin, Peter. "IBM System 360, Model 30." *Facts and stories about Antique (lonesome) Computers.* December 2009. http://ed-thelen.org/comp-hist/ibm-360-30.html (accessed March 30, 2013).

Hendricks, E. C., and T. C. Hartmann. "Evolution of a virtual machine subsystem." *IBM Systems Journal* (IBM Corporation) 18, no. 1 (1979): 111-142.

Levey, Steven. *Hackers: Heroes of the Computer Revolution - 25th Anniversary Edition.* Sebastapol: O'Reilly Media, Inc., 2010.

MacKinnon, R. A. "The changing virtual machine environment: Interfaces to real hardware, virtual hardware, and other virtual machines." *IBM Systems Journal* (IBM Corporation) 18, no. 1 (1979): 18-46.

McCarthy, John. "Reminiscences on the History of Time Sharing." Unpublished Paper, Department of Computer Science, Stanford University, 1983.

Meyer, R. A., and L. H. Seawright. "A virtual machine time-sharing system." *IBM Systems Journal*, 1970: 199-218.

MIT Science Reporter. "Timesharing: A Solution to Comptuer Bottlenecks." *Youtube.* May 9, 1963. https://www.youtube.com/watch?v=Q07PhW5sCEk (accessed March 30, 2013).

Orr, Ken. "How to manage a large-scale IT project." *Computerworld.* October 28, 2004. http://www.computerworld.com/s/article/97019/How_to_manage_a_large_scale_IT_project (accessed March 30, 2013).

Parmelee, R. P., T. I. Peterson, C. C. Tillman, and D. J. Hatfield. "Virtual storage and virtual machine concepts." *IBM Systems Journal*, 1972: 99-130.

Pugh, Emerson W., Lyle R. Johnson, and John H. Palmer. *IBM's 360 and Early 370 Systems.* Cambridge: MIT Press, 1991.

Ritchie, Dennis M., and Ken Thompson. "The UNIX Time-Sharing System ." *Communications of the ACM* (ACM) 17, no. 7 (July 1974): 365-375.

Rosenblum, Mendel. "The Reincarnation of Virtual Machines." *ACM QUEUE*, July/August 2004: 34-40.

Saltzer, J.H. *MAC-TR-16: CTSS Technical Notes.* Project MAC, MIT, Cambridge: MIT, 1963.

Seawright, L. H., and R. A. MacKinnon. "VM/370-a study of multiplicity and usefulness." *IBM Systems Journal* (IBM Corporation) 18, no. 1 (1979): 4-17.

Srodawa, Ronald J., and Lee A. Bates. "An Efficient Virtual Machine Implementation." *AFIPS National Computer Conference.* New York, NY, 1973.

Thiel, Connie. "Editor's Preface." *IBM Systems Journal* (IBM Corporation) 18, no. 1 (1979): 2-3.

Varian, Melinda. *VM and the VM Community: Past, Present, and Future.* Office of Computing and Information Technology, Princeton University, SHARE Inc., 1997.

Von Vleck, Tom. *The IBM 360/67 and CP/CMS.* December 15, 2010. http://www.multicians.org/thvv/360-67.html (accessed February 17, 2013).

Walden, David, and Tom Van Vleck. *Compatible Time-Sharing System (1961-1973): Fiftieth Anniversary Commemorative Overview* . Washington: IEEE Computer Society, 2011.

Watson, Thomas J. Jr. *Father, Son, and Co.: My Life at IBM and Beyond.* New York: Bantam Books, 1990.

Wise, T.A. "IBM's $5,000,000,000 Gamble." *Fortune*, September 1966: 118.